第 5 章

操作表中的数据

通过前面几章的学习，读者已经清楚如何创建数据库和数据表了。但是，数据表中还没有任何数据，本章将介绍如何向表中添加数据以及管理表中的数据。

本章主要知识点如下：

- 向数据表中添加数据；
- 修改数据表中的数据；
- 删除数据表中的数据。

视频讲解

5.1 添加数据

向数据表中添加数据可以通过 SQL 语句，也可以通过使用 SSMS 完成。

5.1.1 INSERT 语句

使用 SQL 语句向数据表中添加数据前，要先弄清楚添加语句的语法规则。添加语句的关键字是 INSERT。在实际应用中添加语句的语法形式有很多，在本节中介绍一个比较常用的 INSERT 语句的语法形式，具体的语法如下：

```
INSERT INTO table_name(column_name1, column_name2,...)
VALUES(value1,value2,...);
```

- table_name：表名。
- column_name：列名。需要添加值的列名，如果没有指定任何列名，意味着是向表中所有列添加值。
- value：值。向数据表中指定列添加的值，值与表中的列是一一对应的。不仅个数要一致，数据类型也要一致。此外，如果没有指定列名，插入值对应列的顺序就是创建表时列的存放顺序，可以直接在 SSMS 中查看。

5.1.2 向表中的全部列添加值

向表中所有的列同时插入值是一个比较常见的应用。下面就用例 5-1 演示如何应用 INSERT 语句向表中全部列添加值。在演示之前，先了解在本章中要操作的数据表，如表 5.1 所示。

表 5.1 银行账号信息表(bankaccount)

编 号	列 名	数据类型	中文释义
1	id	int	账号
2	name	varchar(20)	姓名
3	password	varchar(20)	密码
4	level	int	等级
5	balance	numeric(9,2)	余额
6	bankcode	int	银行代码
7	idcard	char(18)	身份证号
8	tel	char(11)	手机号
9	addr	varchar(30)	家庭住址
10	remark	varchar(200)	备注

根据表 5.1 的表结构，创建银行账号信息表(bankaccount)的语句如下所示。另外，本章的数据表全部创建在数据库 chapter5 中，读者需要自行创建该数据库。

```
USE chapter5;
CREATE TABLE bankaccount
(
  id         int  PRIMARY KEY,
  name       varchar(20),
  password   varchar(20),
  level      int,
  balance    numeric(9,2),
  bankcode   int,
  idcard     char(18),
  tel        char(11),
  addr       varchar(30),
  remark     varchar(200)
);
```

例 **5-1** 向银行账号信息表(bankaccount)中添加数据。如表 5.2 所示。

表 5.2 银行账号信息表添加的数据

账号	姓名	密码	等级	余额	银行代码	身份证号	手机号	家庭住址	备注
1	张三三	112233	1	1000	1001	11010199001011234	13112345678	海淀区	无

向银行账号信息表(bankaccount)中添加表 5.2 的数据，具体的语法如下：

```
USE chapter5;
INSERT INTO bankaccount
VALUES(1,'张三三','112233',1,1000,1001,'11010199001011234', 13112345678,'海淀区','无');
```

执行上面的语法，向银行账号信息表中添加一条数据了，执行效果如图 5.1 所示。

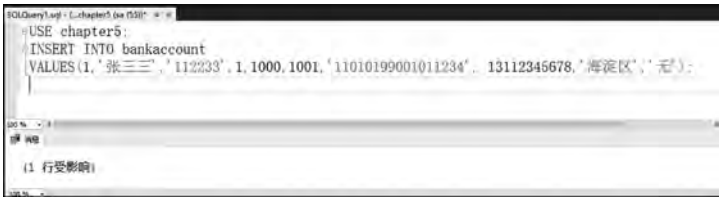


图 5.1　向银行账号信息表(bankaccount)中全部列插入数据

从图 5.1 中可以看到，执行上面的语法后给出的消息是“1 行受影响”，这就意味着有一条记录插入到数据表中了。读者如果想查看数据表中是否有这条数据，用下面的语法就可以了。

```
SELECT * FROM table_name;
```

table_name 是表的名字。将 table_name 换成具体的表名如 bankaccount 就可以查询了，查询效果如图 5.2 所示。

图 5.2　银行账号信息表(bankaccount)中的数据

根据图 5.2 的查询结果可以知道，数据已存放在数据表中了。

5.1.3　给指定列添加值

在实际工作中，添加数据时并不是每次都需要向表中的所有列添加值。这个用 INSERT 语句在插入数据时指定列名就可以。

例 5-2　向银行账号信息表(bankaccount)中插入表 5.3 所示的数据。

表 5.3　银行账号信息表添加的数据

账号	姓名	密码	等级	余额	银行代码	身份证号	手机号	家庭住址	备注
2	李小明	123456	1	5000	1002	210101199202111223	18812345678		

从表 5.3 中可以看出，家庭住址和备注两个列是不需要添加值的，因此添加的 SQL 语句写法如下：

```
USE chapter5;
INSERT INTO bankaccount (id,name,password,level,balance,bankcode,idcard,tel)
VALUES(2,'李小明','123456',1,5000,1002, '210101199202111223', 18812345678);
```

执行上面的语法，向银行账号信息表（bankaccount）中添加一条记录了，执行效果如图 5.3 所示。

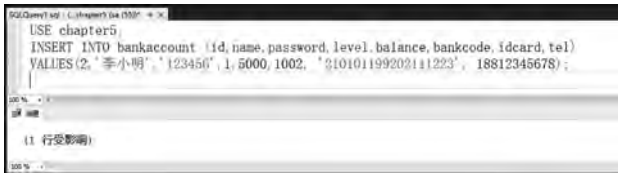


图 5.3　向银行账号信息表指定列插入值

下面利用 SELECT 语句查看插入数据后的效果，如图 5.4 所示。可以看出在例 5-2 中没有添加值的 addr 和 remark 列的值都是 NULL 而不是“”。

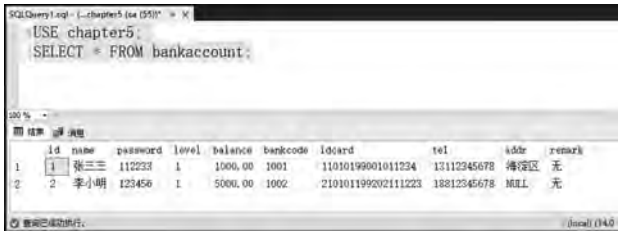


图 5.4　银行账号信息表（bankaccount）中的数据

5.1.4　为标识列添加值

什么是标识列呢？就是在第 3 章中给读者介绍的允许值自动变化的列。标识列只能在整型的列上设置，并且在设置时可以为其指定该列开始的值以及每次的增量。由于标识列的值只能按顺序增加或减少，如果将其中的某一个序号所对应的数据删除掉，系统是不会为其补充该值的。有时有不连续的值输入，需要先将标识列的自动插入属性（IDENTITY_INSERT）设为 ON，允许对标识列中显示值手动插入，具体的设置语法如下：

```
SET IDENTITY_INSERT table_name ON;
```

这里，IDENTITY_INSERT 就是标识列自动插入值的属性，table_name 是表名。如果要将该属性恢复就将上面的语句中 ON 改成 OFF 就可以了。

例 5-3　创建仅包含编号、姓名的用户表，并向标识列中手动添加值。

为了配合标识列的使用，将用户编号列设置为标识列，创建 userinfo 表的语法如下：

```
USE chapter5;
CREATE TABLE userinfo
(
  id   int   PRIMARY KEY IDENTITY(1,1),
  name varchar(20),
);
```

通过上面的语法在数据库 chapter5 中创建数据表 userinfo。

有了用户信息表（userinfo）就可以向该表插入数据了，要添加的数据如表 5.4 所示。

表 5.4　需要添加的数据

账　　号	姓　　名
	小明
5	小晴

从表 5.4 中的数据可以看出，需要向数据表中添加两条记录，第 1 条记录中的用户编号不需要输入，使用标识列自动添加；第 2 条记录的用户编号列手动添加。具体添加语法如下：

```
USE chapter5;
INSERT INTO userinfo
VALUES('小明');
SET IDENTITY_INSERT userinfo ON;
INSERT INTO userinfo (id,name)        -- 必须使用列列表的方式
VALUES(5,'小晴');
SET IDENTITY_INSERT userinfo OFF;
```

执行上面的语法，向用户信息表(userinfo)中添加两条数据，执行效果如图 5.5 所示。

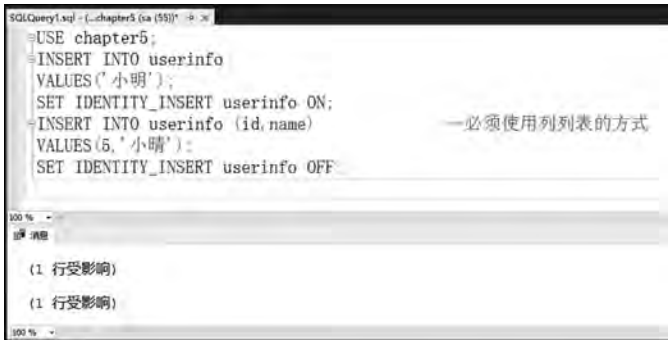


图 5.5　向标识列插入值

向用户信息表(userinfo)中插入两条记录后，查询用户信息表(userinfo)，效果如图 5.6 所示。

从图 5.6 的添加结果可以看出，第 1 条记录中 id 列使用了自增长序列的第 1 个值；第 2 条记录中 id 列使用的是手动添加的值“5”。如果再向用户信息表(userinfo)中插入值时，id 列中的值是“2”还是“6”呢？此时 id 列中的值一定是“6”，因为自增长序列是在该列的最大值基础上增加的。

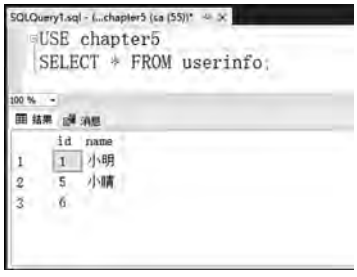


图 5.6　插入 2 条记录后的用户信息表

如果向标识列中添加值，而不将该表的 IDENTITY_INSERT 属性值设为 ON 会出现什么错误呢？下面仍以向用户信息表(userinfo)插入值为列测试如下的语法：

```
USE chapter5;
INSERT INTO userinfo
VALUES(2,'小宁');
```

执行上面的语法会发生什么情况呢？执行效果如图 5.7 所示。

图 5.7 向标识列插入值时出现的错误

请看图 5.7 的错误提示信息，IDENTITY_INSERT 属性值为 ON 时才可以向自增长字段中手动插入值。除了这个提示外，读者还会发现一个信息“仅当使用了列列表”，这个信息的意思是使用 INSERT 语句时，要加上插入列的列名，否则也无法向标识列中插入值。

5.1.5 使用默认值添加数据

在第 4 章中介绍约束时提到过默认值约束的概念，它的作用是当设置默认值约束的列没有插入值时，会自动采用已经设置好的默认值。例如，银行账号信息表(bankaccount)中的备注列，在没有添加具体的备注信息时，默认为“无”。这种情况可以在创建银行账号信息表(bankaccount)时，为其备注字段设置默认值约束，默认值是“无”。在本章已经创建过银行账号信息表(bankaccount)了，请读者使用下面语法给该表的备注列加上默认值“无”。

```
USE chapter5
ALTER TABLE bankaccount
ADD CONSTRAINT df_bankaccount DEFAULT '无' FOR remark;
```

执行上面的语法，为银行账号信息表(bankaccount)的备注字段添加默认值约束了。

例 5-4 向银行账号信息表(bankaccount)中添加如表 5.5 所示的数据。

表 5.5 银行账号信息表添加的数据

账号	姓名	密码	等级	余额	银行代码	身份证号	手机号	家庭住址	备注
3	王胜利	123456	1	5000	1003	210105199202181223	18812345679	沈阳市铁西区	无
4	蒋丽	654321	2	1500	1001	130010199602181223	18612345678	未知	新开户

根据表 5.5 所示的数据，第 1 条数据中的备注字段是“无”，可以在插入的时候利用默认值，具体的语法如下：

```
USE chapter5;
INSERT INTO bankaccount (id,name,password,level,balance,bankcode,idcard,tel,addr)
VALUES(3,'王胜利','123456',1,5000,1003, 210105199202181223, 18812345679,'沈阳市铁西区');
INSERT INTO bankaccount
VALUES(4,'蒋丽',654321,2,1500,1001, 130010199602181223, 18612345678,'未知','新开户');
```

执行上面的语法，为银行账户信息表(bankaccount)添加两条记录，执行效果如图 5.8 所示。

从图 5.8 的效果可以看出，数据已经加入账户信息表中了，那么第 1 条数据中备注字段的值究竟是不是“无”呢？下面查询银行账户信息表看效果，如图 5.9 所示。

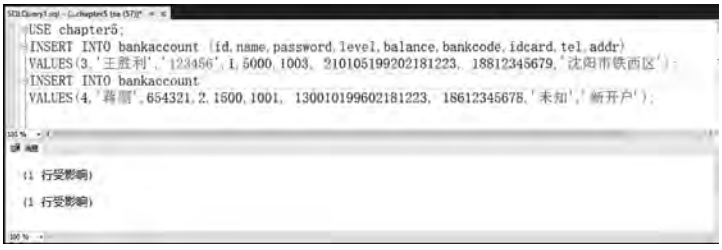
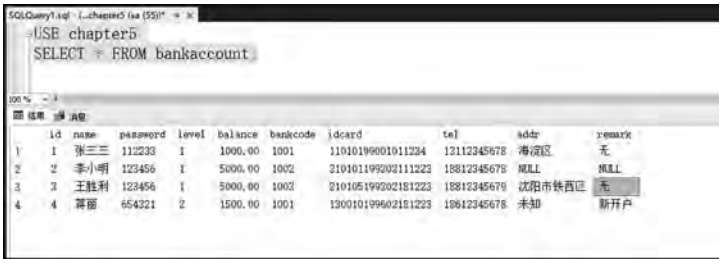


图 5.8　向银行账户信息表(bankaccount)中的默认值列插入值



	id	name	password	level	balance	bankcode	idcard	tel	addr	remark
1	1	张三三	112233	1	1000.00	1001	110101990011111234	12112345678	海淀区	无
2	2	李小明	123456	1	5000.00	1002	210101199202111223	18812345678	NULL	NULL
3	3	王胜利	123456	1	5000.00	1003	210105199202181223	18812345679	沈阳市铁西区	无
4	4	蒋丽	654321	2	1500.00	1001	130010199602181223	18612345678	未知	新开户

图 5.9　银行账户信息表(bankaccount)中的数据

在图 5.9 的 id 列是 3 的这条记录中,remark 列的值是“无”,这就是默认值约束的应用。

5.1.6　复制表中的数据

表和表之间的数据复制要遵守列的个数和数据类型一致性的原则。此外,可以有选择地复制表中一部分列中的数据。将 table_name2 中的数据加入 table_name1 中具体的语法如下:

```
INSERT INTO table_name1(column_name1, column_name2,...)
SELECT column_name_1, column_name_2,...
FROM table_name2
```

- table_name1:插入数据的表。
- column_name1:表中要插入值的列名。
- table_name2:源数据的表。
- column_name_1:table_name2 中的列名。

需要注意的是,在 INSERT INTO 后的列名个数与 SELECT 后的列名个数要一致,数据类型也要兼容。

例 5-5　将表 bankaccount 中的姓名和密码复制到表 bankaccount1 中。

由于表 bankaccount1 不存在,需要先创建 bankaccount1 表,该表仅含有姓名和密码列,创建 bankaccount1 的语法如下:

```
USE chapter5;
CREATE TABLE bankaccount1
(
```

```
  name     varchar(20),
  password varchar(20)
);
```

执行上面的语法，完成 bankaccount1 表的创建。将银行账号表(bankaccount)中的姓名和密码复制到表 bankaccount1 中，具体的语句如下：

```
USE chapter5
INSERT INTO bankaccount1 (name,password)
SELECT name,password
FROM bankaccount;
```

执行上面的语法，将 bankaccount 表中的数据复制到 bankaccount1 表，执行效果如图 5.10 所示。可以看出已经给 bankaccount1 中添加了 4 条记录，查询 bankaccount1 表效果如图 5.11 所示。

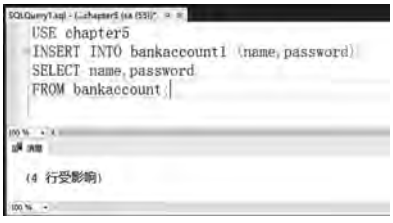


图 5.10　复制 bankaccount 表中的数据

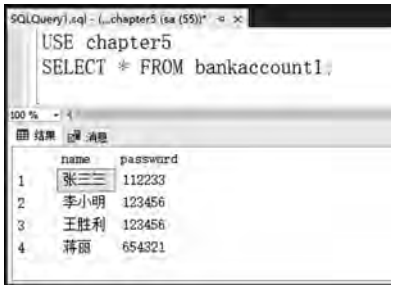


图 5.11　bankaccount1 表中的数据

5.1.7　批量添加

在前面的几个小节中同时向表中添加多条记录，但实际上只是批量地执行 INSERT INTO 语句。同样的操作却要重复地写语句既浪费时间又影响了数据库执行的性能。那么有没有办法只用一条语句就能批量增加数据呢？下面的语法规则就可以完成一次添加多条数据了。

```
INSERT INTO table_name(column_name1, column_name2,...)
VALUES (value1,value2,value3,...),
       (value1,value2,value3,...),
       ...
```

在 VALUES 后面列出要添加的数据，每条记录之间用“,”隔开。

例 5-6　向 bankaccount1 表中添加如表 5.6 中的记录。

表 5.6　bankaccount1 信息表添加的记录

姓　　名	密　　码
Anny	112233
Sophia	213456
Billy	876986

添加记录的语法如下：

```
USE chapter5
INSERT INTO bankaccount1
VALUES ('Anny','112233'),
       ('Sophia','213456'),
       ('Billy','876986');
```

执行上面的语法，一次向 bankaccount1 表添加了 3 条数据，执行效果如图 5.12 所示。查询 bankaccount1 表数据，效果如图 5.13 所示。

图 5.12　向表 bankaccount1 中添加多条记录

图 5.13　批量添加记录后 bankaccount1 表的数据

视频讲解

5.2　修改表中的数据

如果发现数据出现了问题或不符合要求时，不必删除数据，直接修改即可。例如，在注册个人信息后，如果个人邮箱发生变更，只需要更改邮箱而不需要把所有的注册信息全部重新添加。

5.2.1　UPDATE 语句

修改数据表中的数据使用的是 UPDATE 语句，但修改数据表的语句也有很多种形式。这里给出一个通用的形式以便读者学习。其余的形式将在下面的小节中详细讲解，修改数据表的一般语法如下：

```
UPDATE table_name
SET   column_name1 = value1, column_name2 = value2,...
WHERE conditions
```

- table_name：表名。要修改的数据表表名，通常要先打开该数据表所在的数据库。请读者注意，一次只能修改一张表中的数据。
- column_name：列名。需要修改列的名字，value 是给列设置的新值。
- conditions：条件。按条件有选择地更新数据表中的数据。如果省略了该条件，也就是省略了 WHERE 语句，代表修改数据表中的全部记录。

5.2.2 不指定条件修改数据

在上面修改数据表的语法中，省略 WHERE 语句就是不指定条件修改数据，不指定条件即为修改表中的全部数据。

例 5-7 修改银行账号信息表(bankaccount)中的备注信息列(remark)，将其全部修改成“无”。

根据题目要求只是修改银行账号信息表的一列，也就是在 UPDATE 中的 SET 语句后只有一个列，具体语法如下：

```
USE chapter5
UPDATE bankaccount
SET remark = '无';
```

执行上面的语法，将银行账号信息表中 remark 列的值修改成了“无”，具体执行效果如图 5.14 所示。

从图 5.14 中可以看出，通过上面的语句，修改了表中的全部 4 条记录。修改后如何查看结果呢？使用 SELECT 语句，银行账号信息表修改后的效果如图 5.15 所示。

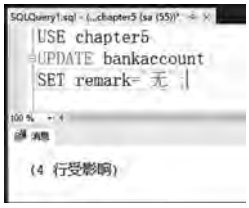


图 5.14 修改银行账号信息表 remark 列

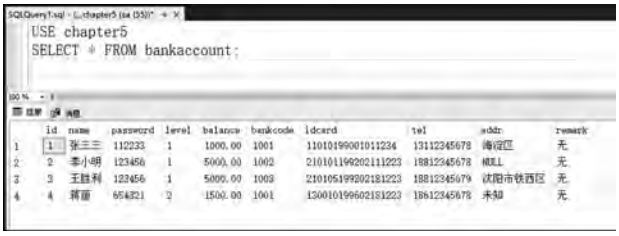


图 5.15 修改银行账号信息表 remark 列后的查询效果

通过图 5.15 可以一目了然，上面的修改语句确实是将银行账号信息表中 remark 列全部修改成了“无”。请读者自己尝试一下，将银行账号信息表的所有的银行编号(bankcode)和等级(level)列分别修改成 1001 和 1。

5.2.3 按指定条件修改数据

所谓按指定条件修改数据，就是在 UPDATE 语句中使用 WHERE 子句指定条件，按条件有选择地修改数据表中的数据。例如，将银行账号信息中银行代码是 1001 的都修改成 1011，将等级是 1 级的都修改成 2 级等。

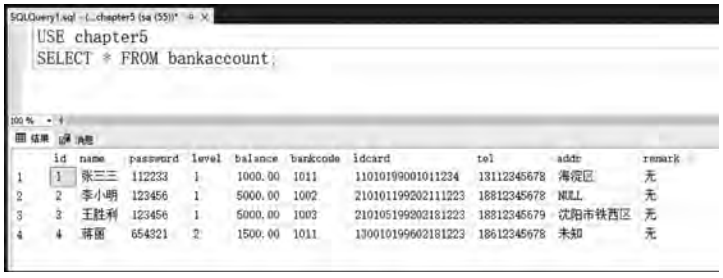
例 5-8 将银行账号信息表(bankaccount)中银行代码是 1001 的更改为 1011。

根据题目要求，修改银行账号信息表的条件只有一个，并且要修改的列只有账户的银行代码列，具体的修改语法如下：

```
USE chapter5;
UPDATE bankaccount
SET bankcode = 1011
WHERE bankcode = 1001;
```

通过上面的语法将银行账号信息表中银行代码是 1001 的全部修改成 1011 了，执行效果如图 5.16 所示。可以看出，执行上面的修改语句后，修改了账号信息表中的 4 条记录。查询银行账号信息表，效果如图 5.17 所示。

图 5.16　修改银行账号信息表中银行代码是 1001 的记录



	id	name	password	level	balance	bankcode	idcard	tel	addr	remark
1	1	张三三	112233	1	1000.00	1011	11010199001011234	13112345678	海淀区	无
2	2	李小明	123456	1	5000.00	1002	210101199202111223	18812345678	NULL	无
3	3	王胜利	123456	1	5000.00	1003	210105199202181223	18812345679	沈阳市铁西区	无
4	4	蒋国	654321	2	1500.00	1011	130010199602181223	18612345678	未知	无

图 5.17　修改银行账号信息表的银行代码是 1001 后的效果

在银行账号信息表中现在一共有 2 条记录的银行代码都是 1011，如何知道修改语句执行成功了呢？可以换个角度看，如果银行账号信息表中没有银行代码是 1001 的记录就说明修改成功了。

5.2.4　修改前 N 条数据

如果在修改数据时，想完成把账号等级是 2 的记录只修改 1 条，应该怎么办呢？仔细想一下，用前面给出的 UPDATE 语句形式是无法完成的。修改的方法是用 TOP 关键字，具体的语法如下：

```
UPDATE TOP (n) table_name
SET column_name1 = value1, column_name2 = value2,...
WHERE condition;
```

TOP (n)中的 n 是指前几条记录，是一个整数，其余的语法都可参考前面的 UPDATE 语句。

例 5-9　修改银行账号信息表(bankaccount)中等级是 1 的前两条记录，将其备注信息修改成“这是刚修改过的”。

根据题目要求，该语句应该用 WHERE 子句限制条件，并用 TOP(2)确定修改两条记录，修改语法如下：

```
USE chapter5;
UPDATE TOP(2) bankaccount
SET remark = '这是刚修改过的'
WHERE level = 1;
```

执行上面的语法，将 level 为 1 的前两条记录修改了，效果如图 5.18 所示。

为了让读者直观地看到修改的效果，特地将 remark 字段修改成了“这是刚修改过的”。查询修改后银行账号信息表，效果如图 5.19 所示。

图 5.18　修改银行账号信息表中等级为 1 的前两条记录

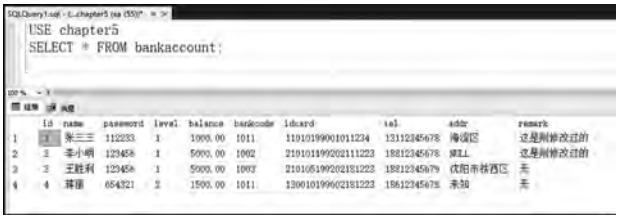


图 5.19　修改银行账号信息表后的效果

5.2.5　根据其他表的数据更新表

上面几个修改数据表的例子都是通过自己指定数据修改的，如果要修改的数据在数据库中的另一张表中已经存在了，可不可以直接复制过来使用呢？当然是可以的，但是不像在 Excel 表格中复制数据那么简单，完成从另一个表复制数据的语法如下：

```
UPDATE table_name
SETcolumn_name1 = table1_name.column_name1,
   column_name2 = table1_name.column_name2,...
FROM table1_name
WHERE conditions
```

- table_name：表名。要更新的数据表名称。
- column_name1：列名。要修改数据表中的列名。
- table1_name：表名。要复制数据的数据表名称。
- table1_name.column_name1：列名。要复制数据的列，记住，一定要在列名前面加上表名。
- conditions：条件。它是指按什么条件复制表中的数据。

按照上面的语法规则可以在修改数据表时使用其他表中的数据了。下面通过例 5-9 进行练习。在练习之前先创建一张数据表并存入一些数据，以便复制数据使用。创建一张银行卡等级信息表，表结构如表 5.7 所示。

表 5.7　银行卡等级信息表(cardlevel)

序　号	字 段 名	数 据 类 型	描　述
1	id	int	等级编号
2	level	int	等级
3	minlimit	numeric(10,2)	存款额最小值
4	maxlimit	numeric(10,2)	存款额最大值

上面的银行卡等级信息表主要描述了每种等级对应的存款额要求，根据表结构创建数据表的语法如下：

```
USE chapter5;
CREATE TABLE cardlevel
(
```

```
    id         int  PRIMARY KEY IDENTITY(1,1),
    level      int,
    minlimit   numeric(10, 2),
    maxlimit   numeric(10, 2)
);
```

执行创建数据表的语法后，再为其添加如表 5.8 中的数据，要复制的数据来源创建好了。

表 5.8　银行卡等级信息表添加的数据

等级编号(id)	等级(level)	存款额最小值(minlimit)	存款额最大值(maxlimit)
1	1	0	5000
2	2	5001	20000
3	3	20001	100000

批量向数据表中添加数据的具体语法如下：

```
INSERT INTO cardlevel
VALUES(1,0,5000),(2,5001,20000),(3,200001,100000);
```

执行上面的语法后，银行账号等级信息表中就有了 3 条数据了。

例 5-10　将银行账号信息表中的等级信息按照银行卡等级信息表中每个等级对应的存款额度进行更新。

根据题目要求，要更新银行卡账号信息表中的等级，更新语法如下：

```
USE chapter5;
UPDATE bankaccount
SET bankaccount.level =  cardlevel.level
FROM cardlevel
WHERE bankaccount.balance >= cardlevel.minlimit AND bankaccount.  balance < cardlevel.maxlimit;
```

执行上面的语法，将银行账号信息表中的银行卡等级根据卡中存款额进行更新，执行效果如图 5.20 所示。

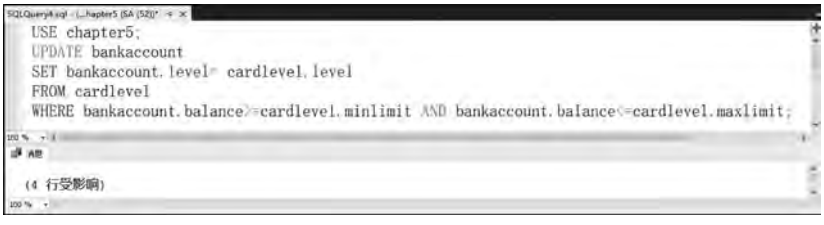
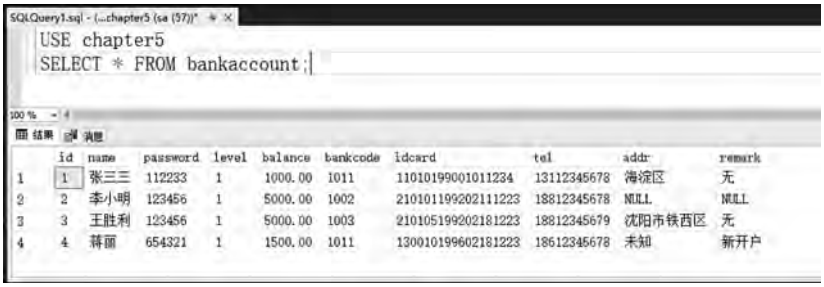


图 5.20　更新银行账号信息表中的等级

从图 5.20 所示的执行效果中可以看出，银行账号信息表中的 4 行数据已经被更新了。更新后的银行账号信息表中的数据如图 5.21 所示，可以看出银行账号信息表中的等级已经按照银行卡等级中的额度范围进行了更新。



	id	name	password	level	balance	bankcode	idcard	tel	addr	remark
1	1	张三三	112233	1	1000.00	1011	11010199001011234	13112345678	海淀区	无
2	2	李小明	123456	1	5000.00	1002	210101199202111223	18812345678	NULL	NULL
3	3	王胜利	123456	1	5000.00	1003	210105199202181223	18812345679	沈阳市铁西区	无
4	4	蒋丽	654321	1	1500.00	1011	130010199602181223	18612345678	未知	新开户

图 5.21　银行账号信息表更新后的效果

5.3　使用 DELETE 语句删除表中的数据

视频讲解

如果某些数据已经没有作用就可以将其删除了。但是，删除的数据不容易恢复，注意备份。

5.3.1　DELETE 语句

在删除数据表中的数据前，如果不能确定这个数据以后是否还会有用，一定要对其数据先进行备份，删除数据表中的数据使用的语法很简单，一般的语法形式如下：

```
DELETE FROM table_name
WHERE conditions;
```

- table_name：表名。要删除数据的数据表名称。
- conditions：条件。按照指定条件删除数据表中的数据，如果没有指定删除条件删除表中的全部数据。

此外，在上面的语法中也可以将 DELETE 后面的 FROM 关键字省略。

5.3.2　删除表中的全部记录

删除表中的全部数据，就是在使用删除语句时不加 WHERE 子句，下面就用例 5-11 演示如何删除表中的全部数据。

例 5-11　将银行卡等级信息表中的数据全部删除。

根据题目要求，删除语法如下：

```
USE chapter5;
DELETE FROM cardlevel;
```

执行上面的语法，银行卡等级信息表中的记录消失了，效果如图 5.22 所示。可以看到已经有 3 行记录被删除了，删除后剩下一张空表，效果如图 5.23 所示。

从图 5.23 中可以看出，虽然将银行卡等级信息表中的数据删掉了，但是表的结构还是保留的。

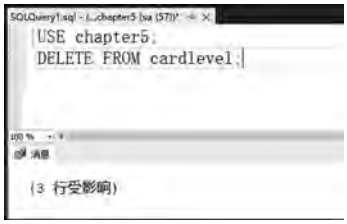


图 5.22　删除银行卡等级信息表中的所有数据

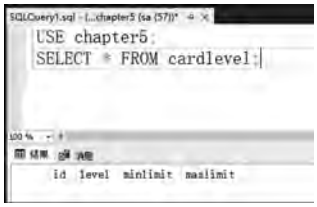


图 5.23　清空后的银行卡等级信息表

5.3.3　按条件删除记录

在实际应用中，删除表中全部记录的操作是不太常用的，经常使用的是按条件删除数据，在删除表中数据的语法中加 WHERE 子句。

例 5-12　将银行账号信息表中等级是 1 的记录删除。

根据题目要求，只删除等级是 1 的记录，删除语法如下：

```
USE chapter5;
DELETE FROM bankaccount
WHERE level = 1;
```

执行上面的语法，将银行账号信息表中等级是 1 的记录删掉了，效果如图 5.24 所示。

从图 5.24 中可以看出，影响了表中的 4 条记录，也就是原来表中有 4 条记录等级是 1。删除后查看银行账号信息表，效果如图 5.25 所示。

图 5.24　删除等级是 1 的记录

图 5.25　删除等级是 1 的记录后的效果

读者在图 5.25 中发现所有的记录都被删除了。

5.3.4　删除前 N 条数据

先请读者思考一个问题，如果想删除两条等级是 2 的银行账户信息，怎么办呢？之前学习的 DELETE 语句的语法形式不能够满足要求，除非在银行账号信息表中只有两条等级是 2 的记录，才可以通过在 DELETE 语句的 WHERE 子句里设置条件删除。其他情况请看如下的语法学习如何删除数据。

```
DELETE TOP(N) table_name
WHERE conditions;
```

TOP(N)就是用来指定删除前 N 条记录的。TOP(2)就是指前 2 条记录。

下面通过例 5-13 练习如何删除前 N 条数据。

例 5-13 从银行账户信息表中删除 1 条等级是 1 的银行账户信息。

根据题目要求，删除 1 条记录使用 TOP(1)就可以了，删除语法如下：

```
USE chapter5;
DELETE TOP(1) bankaccount
WHERE level = 1;
```

执行上面的语法，即仅删除 1 条等级是 1 的记录。

5.3.5 使用 TRUNCATE TABLE 语句清空表中的数据

前面已经讲到在清空表中的数据时，可以使用 DELETE 语句完成。实际上，除了使用 DELETE 语句外，还可以使用 TRUNCATE TABLE 语句清空表中的数据。TRUNACTE TABLE 语句的语法很简单，只需要在其后面加上表名就可以了。具体的语法形式如下所示：

```
TRUNCATE TABLE table_name;
```

table_name 就是表名，同样在删除之前还要打开该表所在的数据库。

例 5-14 使用 TRUNCATE TABLE 语句删除银行账号信息表(bankaccount)中的数据。

根据题目的要求，删除银行账号信息表中的数据，语法如下：

```
USE chapter5;
TRUNCATE TABLE bankaccount;
```

执行上面的语法，将银行账号信息表中的数据全部删除，效果如图 5.26 所示。

从图 5.26 的执行效果可以看出，在执行了 TRUNCATE TABLE 语句后并没有像执行 DELETE 语句时出现“影响几行”的消息提示，而是出现“命令已成功完成”的消息提示了。很显然，TRUNCATE TABLE 语句和 DELETE 语句不是一种类型的语句。实际上，TRUNCATE TABLE 属于数据定义语言，与 CREATE、DROP、ALTER 是一类的。

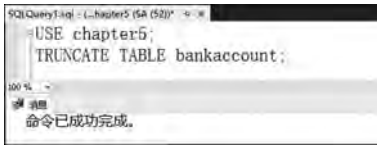


图 5.26 删除银行账号信息表中的全部数据

通过上面的示例，相信读者已经了解了 TRUNCATE TABLE 的基本用法。那么，TRUNCATE TABLE 与 DELETE 删除表中的数据有什么区别呢？主要区别有两个，具体说明如下：

(1) 使用 TRUNCATE TABLE 删除数据速度较快。使用 DELETE 语句删除数据时，需要把删除的信息写入到事务日志文件中，这样能够编译恢复删除的数据。而使用 TRUNCATE TABLE 语句删除数据，是不会将删除的信息写入事务日志文件的。因此，使用 TRUNCATE TABLE 删除数据的速度较快。当表中的数据不再需要恢复时，可以使用 TRUNCATE TABLE 语句完成删除操作。

(2) 使用 TRUNCATE TABLE 删除数据后标识列重新编号。使用 TRUNCATE TABLE

语句清空表中的所有记录后，表中的标识列会重新编号；而使用 DELELE 删除表中的全部记录后，表中的标识列会继续变化。为了验证 TRUNCATE TABLE 与 DELETE 删除数据后对标识列的影响，新创建一张数据表 test，表结构如表 5.9 所示。

创建好 test 表后，向表中添加如表 5.10 的数据。

表 5.9　test

序　号	字 段 名	数 据 类 型	描　述
1	id	int	编号
2	name	varchar	名称

表 5.10　test 表添加的数据

编号(id)	名称(name)
1	桌子
2	椅子
3	书

创建 test 表并添加数据的语法如下：

```
USE chapter5;
CREATE TABLE test
(
  id     int  IDENTITY(1,1),
  name   varchar(20)
);
INSERT INTO test VALUES('桌子'),('椅子'),('书');
```

执行上面的语法，查询 test 表中的数据，如图 5.27 所示。

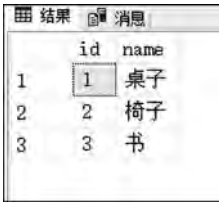


图 5.27　test 表中的数据

下面分别使用 TRUNCATE TABLE 语句和 DELETE 语句删除 test 中的数据。

(1) 使用 TRUNCATE TABLE 语句删除 test 中的数据。删除之前在图 5.27 中，id 是自增长列并且序号已经变成 3 了。删除语法如下：

```
USE chapter5;
TRUNCATE TABLE test;
```

执行上面的语法，test 表中没有数据了。下面向 test 表中再插入一条数据，语法如下：

```
USE chapter5;
INSERT INTO test VALUES('本');
```

查询 test 表中的数据，看看 id 究竟是几，如图 5.28 所示。

从图 5.28 中可以看到，id 值是 1，也就是说使用 TRUNCATE 语句删除数据后，自增长列是重新计数的。

(2) 使用 DELETE 语句删除 test 表中的数据。当前 test 表中只有一条数据，id 为 1。下面使用 DELETE 语句将其删除，语法如下：

```
USE chapter5;
DELETE test;
```

执行上面的语法，test 表中的数据也被删除了。下面同样再向 test 表中添加 1 条记录，

看看 id 的变化，添加数据的语法如下：

```
USE chapter5;
INSERT INTO test VALUES('书包');
```

再查看 test 表，看看 id 是多少了，如图 5.29 所示。

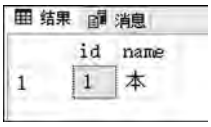


图 5.28　使用 TRUNCATE 删除数据后标识列的变化

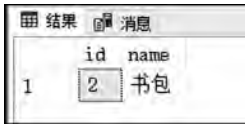


图 5.29　使用 DELETE 删除数据后标识列的变化

这回看到 id 的值是 2，也就是说，通过 DELETE 删除数据后，标识列的序号是在原来的基础上继续增加的。

5.4　使用 SSMS 操作数据表

前面学习了使用 SQL 语句对数据表中数据进行添加、修改以及删除的操作，这些语句需要记住他们才能使用。现在学习一个简单的方法，使用 SSMS 操作数据表。

SSMS 是一个用鼠标操作的友好界面，使用 SSMS 操作表中的数据最能够体现它的便利了。下面通过例 5-15 演示如何在 SSMS 中添加、修改以及删除数据。

例 5-15　在 SSMS 中对银行账号信息表做如下操作。

(1) 向数据表中添加如表 5.2 的数据。

(2) 修改编号是 1 的记录，将其余额修改成 2000。

(3) 删除编号是 1 的记录。

无论对银行账号信息表做添加、修改还是删除操作，都需要在银行账号信息表的表编辑界面中完成。在 SSMS 的“对象资源管理器”中，展开 chapter5 数据库，右击该数据库中的表 bankaccount，在弹出的快捷菜单中选择“编辑前 200 行”选项，即可见到银行账号信息表的编辑界面，如图 5.30 所示。

图 5.30　银行账号信息表的编辑界面

本题的 3 个小问，都可以在图 5.30 所示的界面中完成。下面分别讲解如下。

(1) 添加表 5.2 的数据，就像在 Excel 表中输入信息一样。录入表 5.2 的信息后，效果如图 5.31 所示。

添加数据后如何保存呢？在 SSMS 中，为表添加数据不用刻意地去保存数据，只需要把光标移动到下一行，就保存了上一行的数据。

(2) 将 id 是 1 的记录余额改成 2000。在图 5.31 所示的界面中，直接将 id 为 1 的列所对应的 balance 列改成 2000 就可以了，效果如图 5.32 所示。

DESKTOP-GF03EE...dbo.bankaccount

	id	name	password	level	balance	bankcode	idcard	tel	addr	remark
	1	张三三	112233	1	1000.00	1001	11010199001...	13112345678	海淀区	无
**	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 5.31　向 bankaccount 表中添加数据

DESKTOP-GF03EE...dbo.bankaccount

	id	name	password	level	balance	bankcode	idcard	tel	addr	remark
	1	张三三	112233	1	2000.00	1001	11010199001...	13112345678	海淀区	无
**	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 5.32　修改 id 为 1 的列

从图 5.32 中可以看出，当前银行账号信息表的状态是“单元格已修改”，修改数据后，仍然将光标移动到其他单元格中，就可以保存该数据了。

（3）删除数据会稍微复杂一些。删除编号是 1 的记录，需要先单击要删除的记录使其处于选中状态，然后右击该记录，在弹出的快捷菜单中选择“删除”选项，弹出如图 5.33 所示的对话框。

图 5.33　删除提示

在图 5.33 所示的界面中单击“是”按钮，即可将所选的记录删除了。

说明：如果要删除多条记录，不用一条一条地选择，只需要使用 Shift 或 Ctrl 键选中要删除的记录，然后一起删除就可以了。

5.5　本章小结

本章主要讲解了如何使用 SQL 和 SSMS 向数据表中添加、修改以及删除数据。在添加数据部分，着重讲解了如何给标识列添加值、复制表中的数据以及批量添加数据；在修改数据部分，着重讲解了如何修改前 N 条记录以及如何在修改时使用其他表中的数据；在删除数据部分，着重讲解了如何删除 N 条记录，同时也讲解了 TRUNCATE TABLE 语句与 DELETE 语句的区别。希望读者通过本章的学习，能够随意地使用 SQL 语句操作数据表中的数据，而且，SSMS 也得会用。

5.6　本章习题

一、填空题

1. 删除表中全部数据，可以使用_____语句。

2. 修改表中前 N 项数据需要使用的关键字是_____。

3. 未插入值的列，列中的值为_____。

二、选择题

1. 下面对向数据表中插入数据的描述正确的是(　　)。

A. 可以一次向表中的所有字段插入数据

B. 可以根据条件向表中的字段插入数据

C. 可以一次向表中插入多条数据

D. 以上都对

2. 下面对修改数据表中的数据描述正确的是(　　)。

A. 一次只能修改表中的一条数据

B. 可以指定修改前 N 条数据

C. 不能修改主键字段

D. 以上都不对

3. 下面对删除数据表中的数据描述正确的是(　　)。

A. 使用 DELETE 只能删除表中的全部数据

B. 使用 DELETE 可以删除 1 条或多条数据

C. 使用 TRUNCATE 语句也能删除 1 条或多条数据

D. 以上都不对

三、问答题

1. INSERT 语句的基本语法形式是什么？

2. UPDATE 语句的基本语法形式是什么？

3. DELETE 与 TRUNCATE 的区别是什么？

四、操作题

使用表 5.1 的银行账号信息表，完成如下 SQL 语句的编写。

(1) 向银行账号信息表中任意添加 5 条数据。

(2) 将银行账号信息表中前三条记录中账号等级加 1。

(3) 删除所有账号等级为 1 的账号信息。